Orchestrating a brighter world NEC

経済産業省　大臣官房 デジタル・トランスフォーメーション室 御中

令和４年度 効率的なデータ整備・データ可視化の導入実施・調査事業

調査報告書

2023年3月31日

NECソリューションイノベータ株式会社
官公ソリューション事業部

目次

1. 本事業の全体像
   - 1.1 本事業の実施作業
   - 1.2 本事業の目的
   - 1.3 本事業の実施スケジュール
2. 本事業の効果設定
   - 2.1 現状課題と実施事項の設定
   - 2.2 グランドデザイン
   - 2.3 実証結果
   - 2.4 更なる効果創出のための必要事項
3. 実証作業
   - 3.1 実証に用いるデータ利活用基盤
   - 3.2 データ整備方法
   - 3.3 データ可視化方法
4. データ整備・可視化方法の標準化検討
   - 4.1 接続・収集方式における作業
   - 4.2 編集方式における作業
   - 4.3 加工方式における作業
   - 4.4 データ加工処理概要
   - 4.5 セキュアコーディング
   - 4.6 グラフ作成
   - 4.7 定期的な実行

1. 本事業の全体像

1.1 本事業の実施作業
1.2 本事業の目的
1.3 本事業の実施スケジュール

1.1 本事業の実施作業

本事業での実施内容とそれらに該当する本報告書での該当箇所は以下の通り。

実施内容(仕様書記載事項)		本報告書での記載内容	該当頁
データ整備方法の実証	① 接続・収集方式	• オープンデータへの接続・収集方式を実証	P.16～17
	② 編集・加工方式	• 内外データの編集・加工方式を実証	P.18～19
	③ 保存方式	• 編集・加工されたデータの保存方式を実証	P.20
	④ 基盤としての設定検討	• 貴省保有のデータ可視化ツールを用いてデータを利活用するために必要な設定検討	P.21～22
データ可視化方法の実証		• 貴省保有のデータ可視化ツールを用いての可視化手法を実証	P.23～26
データ整備・可視化方法の標準化検討		• 上記により洗い出されたベストプラクティスを標準化資料として取りまとめ	P.27～39

1.2 本事業の目的

本事業の目的は、「デジタル・ガバメントの実現に向けて、効果的かつ効率的なデータの管理・活用方法を検討すること」であり、より本格化する「データの利活用」を推進するため、「省内の業務において蓄積したデータの整備・可視化方法を確立することにより、当該データ／グラフを効果的な政策立案に活用していくこと」を目指すものである。

本事業の背景サマリ

■政策サービス対象である事業者等の満足度向上

✓ データを政策立案に活用し政策の効果を高める

■部局内データの有機的結合による相乗効果の探求

✓ 縦割りで蓄積されたデータを有機的に結合することで、政策分析の効果を高める

本事業の目的

省内の業務において蓄積したデータの整備・可視化方法を確立する

1. 本事業の全体像

1.3 本事業の実施スケジュール

- 本事業の前半においては、1部局が行っている経済指標の調査レポートに関して、一連のデータ整備・可視化方法を実装し、本事業の実証のひな型を構築
- 後半においては、前半で構築したひな形が水平展開可能かを別の部局へ展開し、ベストプラクティスとしての妥当性を検証
- これらの実証を通じて、標準化すべき事項を検討の上、調査報告書として取りまとめた

2. 本事業の効果設定

2.1 現状課題と実施事項の設定
2.2 グランドデザイン
2.3 実証結果
2.4 更なる効果創出のための必要事項

2.1 現状課題と実施事項の設定

現状課題

各業界の動向把握や政策のモニタリングなどにおけるデータの可視化・共有コストが肥大化している

想定される原因

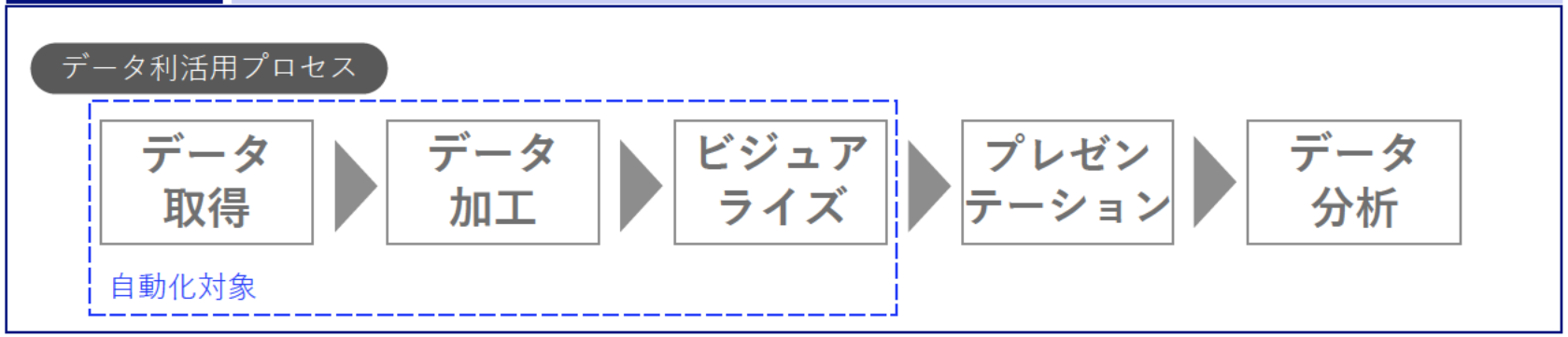
① **多数のデータを扱う**
数多のデータソースからの収集、保管にリソースを消費

② **定期的な手動データ加工**
数式誤り等を防ぐ、人間系での労力が多大

③ **配布によるデータ共有**
データとのインタラクティブな対話の機会を喪失

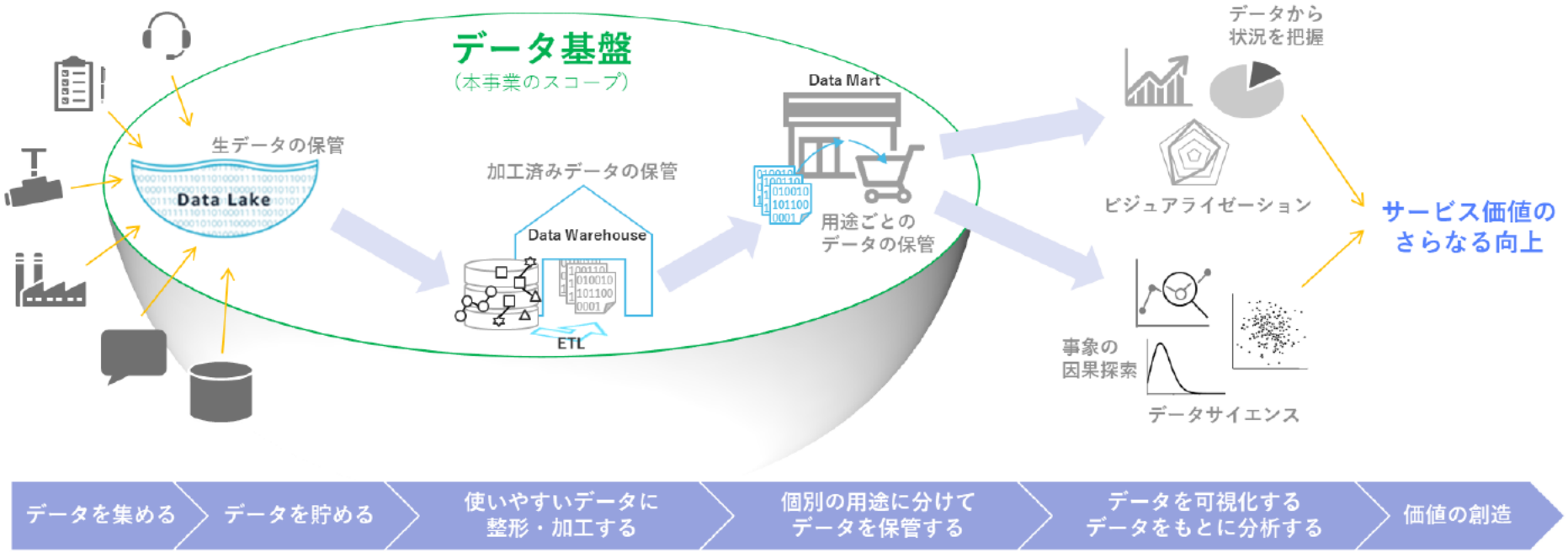
本事業の実施事項

手動の工程の排除による データ可視化・共有コスト低減



2.2 グランドデザイン

分析などに利用するデータを蓄積し、必要に応じて取り出すことができる処理システム群として**データ基盤**を構築。



2.2 グランドデザイン

Conceptは、効率的なデータ利活用を手助けするData Market

効率的なデータ収集

様々なサイトから統一的なフレームワークとスケジュールでデータを収集する

安全なデータ保管

形式外データの混入を防ぎ、鮮度の高いデータ保管を実現する

使いやすい形式へのデータ加工

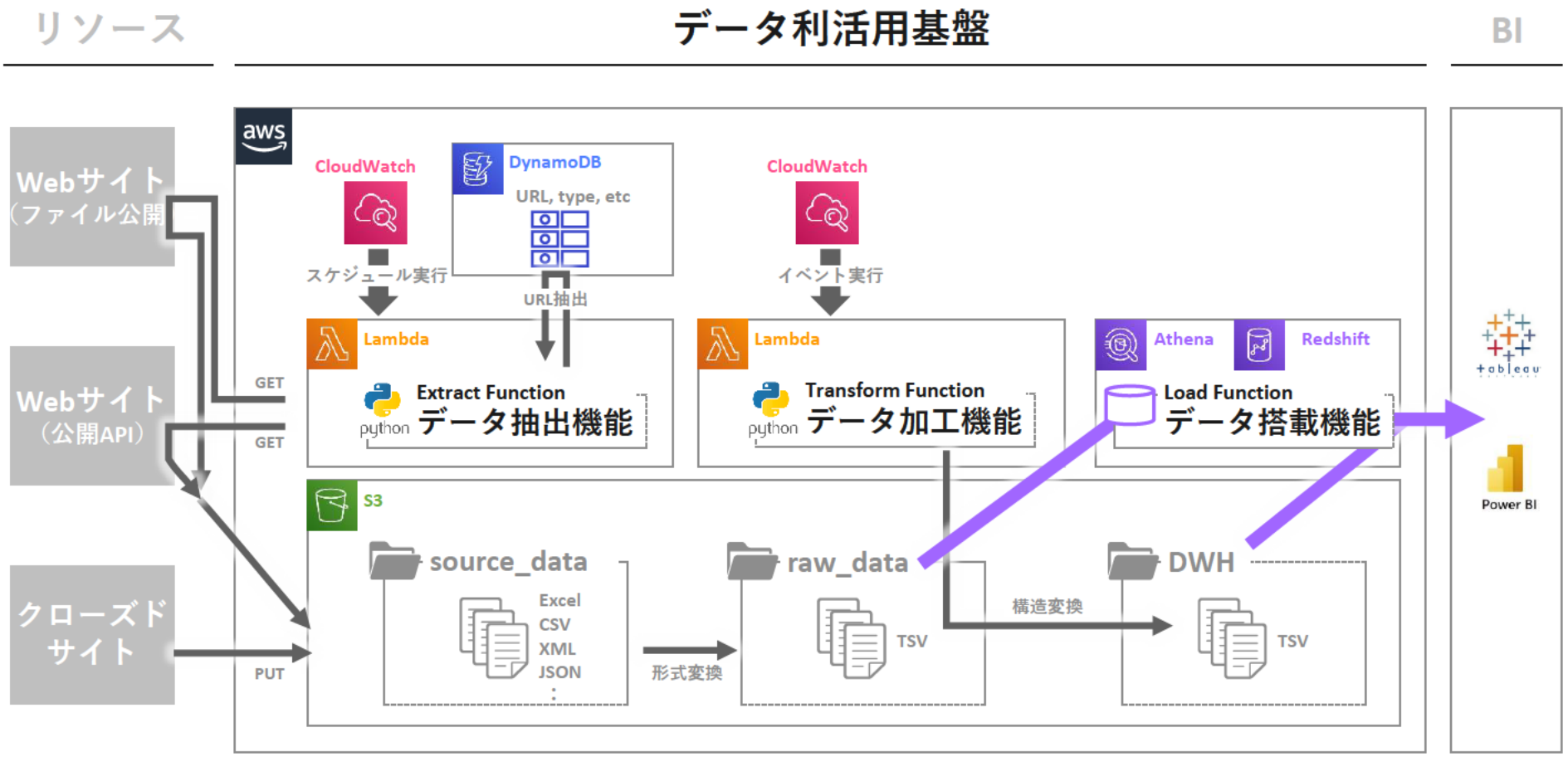
様々なビジュアライゼーションに対応する加工済みデータを生成する

使える形でのデータ提供

カタログ、レシピをセットに、簡易なインタフェースでデータを提供する

2.3 実証結果

構成詳細図

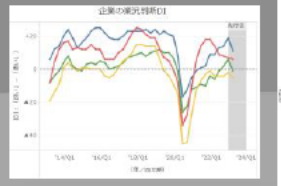


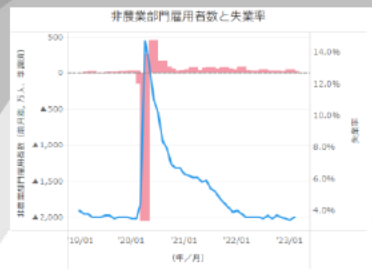
2.3 実証結果

国内/海外の経済指標データ

313種を搭載し60のグラフ化

2023年2月末時点





2.4 更なる効果創出のための検討課題

見せる から 使う へ

共通して「見る」「使う」データの拡充

経済産業省職員が共通して見るべきデータを実装

職員からの アクセシビリティ向上

アクセスしやすいデータポータルを入り口に

レシピのセット提供による 職員リテラシー向上

メタデータの拡充による使い方の理解促進の橋頭堡に

ソリューション例

可視化・分析

可視化・分析の作業に注力可能

リアルタイム分析

更新頻度が高いデータを日次処理ではなく、すぐに利用

予測分析

意思決定の参考として
過去～現在の事象のデータを利用

アプリケーション構築

データ取得先・形式の一元化による工数削減

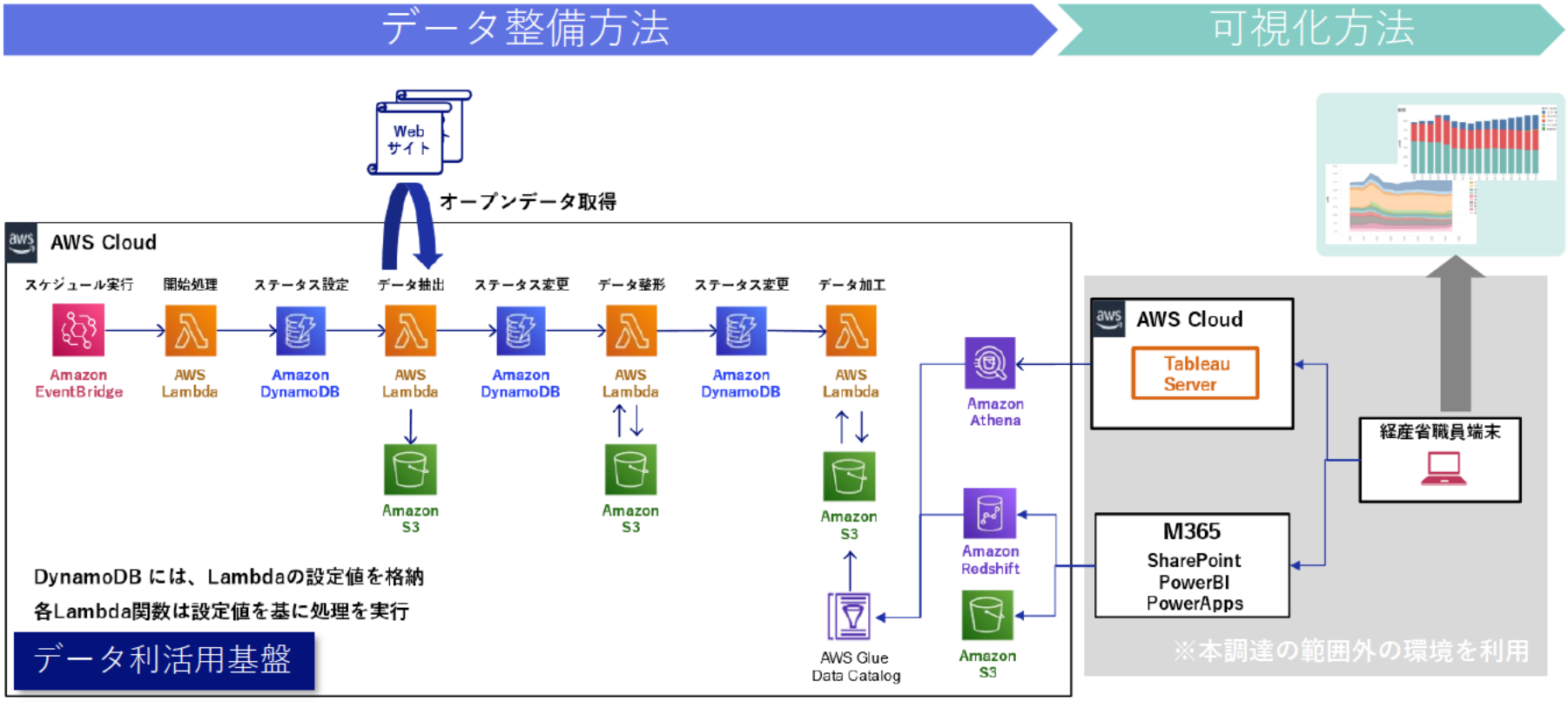
3. 実証作業

3.1 実証に用いるデータ利活用基盤
3.2 データ整備方法
3.3 データ可視化方法

3. 実証作業

3.1 実証に用いたデータ利活用基盤

以下の構成からなるデータ利活用基盤をAWS上に構築し、データ収集から可視化までの一連のプロセスを実現。



3.1 実証に用いたデータ利活用基盤

データ利活用基盤での利用コンポーネントは下記の通り。

接続・収集

- ✓ サーバレスコンピューティングとして、AWS Lambdaを使用
- ✓ Lamda上にPythonを用いて対象サイトからの抽出処理を実装
- ✓ 対象サイトはDynamoDBにて定義情報を管理

編集・加工

- ✓ サーバレスコンピューティングとして、AWS Lambdaを使用
- ✓ Lamda上にPythonを用いて取得データの加工処理を実装
- ✓ 加工方法はDynamoDBにて定義情報を管理

保存

- ✓ オブジェクトストレージとして、Amazon S3を使用
- ✓ 抽出処理、加工処理にて取得したデータの保管に利用
- ✓ 世代管理を行い、データ分析における再現性を確保

蓄積

- ✓ サーバレスクエリとして、Amazon Athenaを使用
- ✓ サーバレスデータウェアハウスとして、Amazon Redshiftを使用
- ✓ AthenaはTableauからのコネクタ、RedshiftはPowerBIからのコネクタとして利用

3.2 データ整備方法

データ整備方法は、大きく「データ抽出」・「データ整形」・「データ加工」の3つの処理に分類することをベストプラクティスとした

接続・収集
データ抽出
- Webサイトからオープンデータを取得し、AWS S3に無加工のファイルを格納する処理

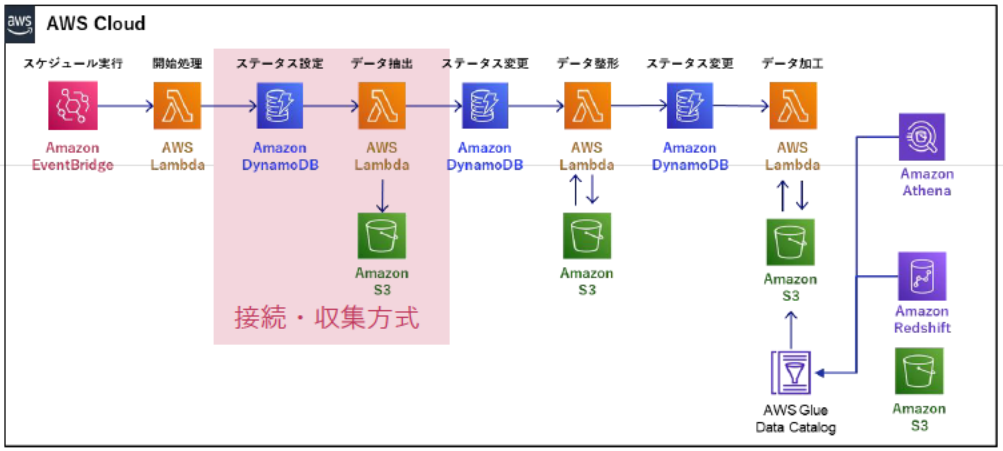
編集
データ整形
- データ抽出でS3に格納したファイルを、加工・可視化しやすいように構造化データに整形する

加工
データ加工
- 整形したデータを、可視化するためのデータとして加工する
- 単位変換(億円→円)等、他のデータセットと組み合わせた際に利活用しやすい形式への変換や、可視化手段での実現が難しい加工を本処理で実施する

3.2 データ整備方法

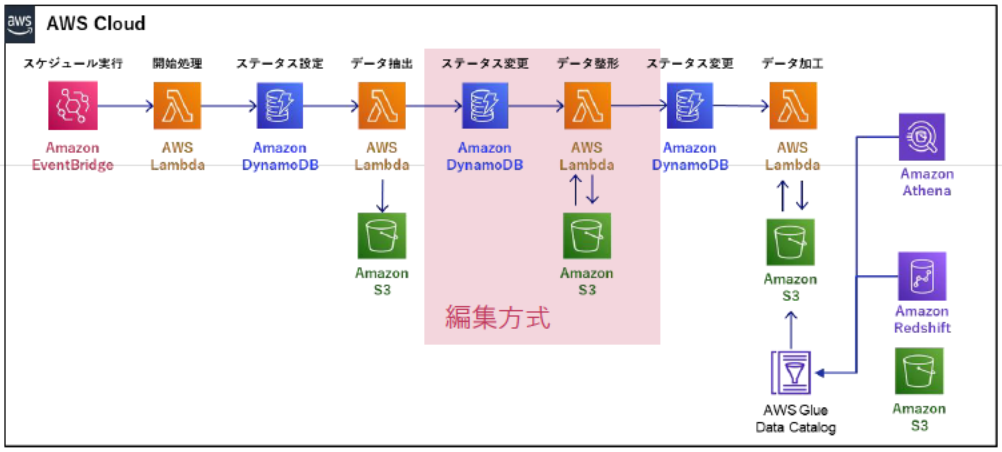
① 接続・収集方式

- データ抽出パターンを5つに分類
- 引数の設定により様々なデータの抽出を可能とする



処理番号	概要
001	対象のURLからファイルをダウンロードする
002	対象のURLからUser-Agentを指定してファイルをダウンロードする
003	対象のURLからPDFをダウンロードし、うち1ページを画像変換する（個別データ対応）
004	対象のURLからPDFをダウンロードし、うち2ページを画像変換し、1枚の画像に合成する（個別データ対応）
005	対象のURLを年月日と文字列でフォーマットし、ファイルをダウンロードする（個別データ対応）

3.2 データ整備方法



②-1 編集方式

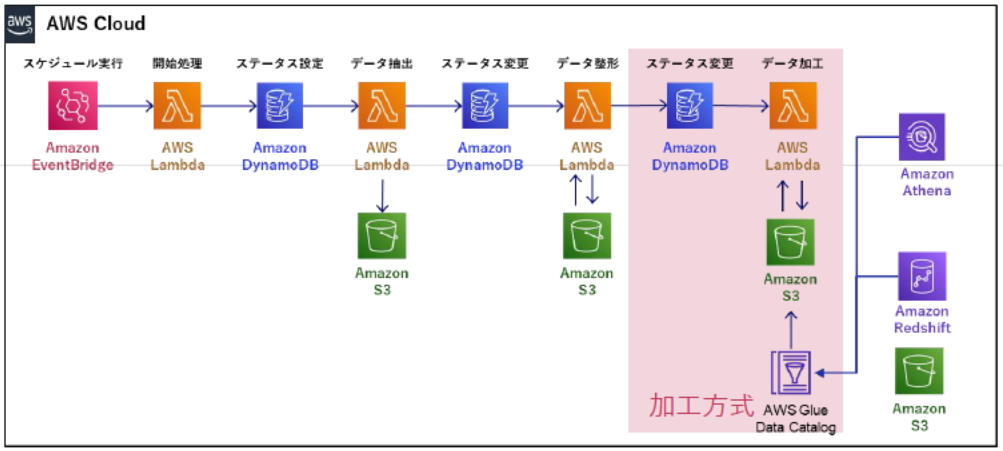
- 取得データをTSV形式の構造化データに整形
- 後続のデータ加工処理の簡易化のため、データ整形処理はデータ個別に関数作成する

No	入力データ拡張子	整形内容
1	CSV（TSV、TXTも含む）	対象のCSVファイルで列番号、行番号を指定し、構造化データに変換する
2	Excel（XLSX、XLSを含む）	対象のExcelファイルでシート名、列番号、行番号を指定し、構造化データに変換する
3	PDF（パターン①）	対象のPDFをExcelに変換し、列番号と行番号を指定し、構造化データに変換する
4	PDF（パターン②）	対象のPDFの指定した画像と、用意しておいた画像群と比較する。 一致する画像を特定後、構造化データに変換する （例：PDFの矢印がどの向きかを特定する）
5	HTML（HTMも含む）	対象のHTMLファイルをHTMLタグを元に、構造化データに変更する
6	XML	対象のXMLファイルを辞書型に変換し、キーを元に、構造化データに変換する

3.2 データ整備方法

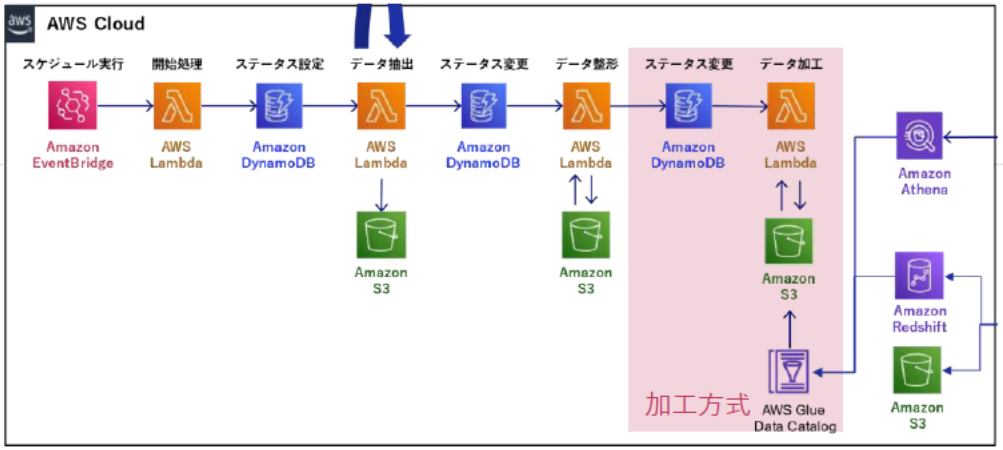
②-2 加工方式

- 整形済TSVファイルを組み合わせ、DWH固有のレイアウトに加工
- DWHを作成後に、コネクタとなるクエリスキーマを作成



処理番号	概要	備考
001	Raw_dataを加工せずにDWHとする	大部分のデータで使用
002	2つのRawDataを加工し、加工済新車販売台数の前年同月比計算処理を行う	データ固有
003	鉱工業、鉱工業以外の品目、データにない品目の計算処理を行う	データ固有
004	各種指標、寄与度の計算を行う	データ固有
005	2つのRawDataを加工し、設備投資の計算を行う	データ固有
006	設備投資の計算を行う	データ固有
007	レポート種別（確速、確報、確々報など）がある場合の共通処理	汎用的に使用できる
008	複数ソースを結合する場合の共通処理（指定したキーで結合）	汎用的に使用できる
009	ヨーロッパ圏、英国、ドイツの景況指数の処理	データ固有
010	複数ソースを結合する。整形したTDBが公表する景気動向調査用処理	データ固有
011	複数ソースを結合する場合の共通処理（複数ソースを縦に結合）	汎用的に使用できる
012	英国雇用データの公表日を1か月後ろ倒しする処理	データ固有
013	マッチング結果のソースを結合・重複排除・項目名を付与する処理（1ページ）	データ固有
014	マッチング結果のソースを結合・重複排除・項目名を付与する処理（2ページ超）	データ固有

3.2 データ整備方法

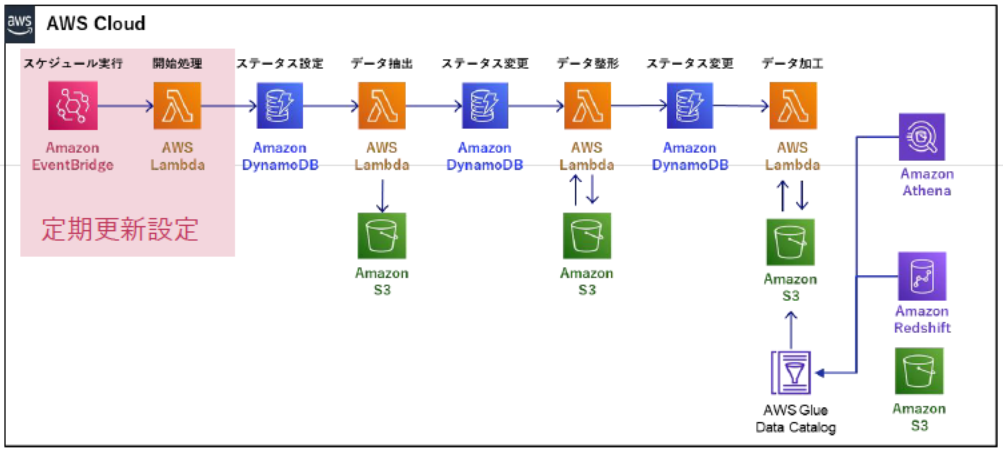


③ 保存方式

- 各処理で生成されたデータはAWS S3に保存する
- データ整形処理、データ加工処理の出力結果はTSV形式とする

No	用途	概要	採番ルール	S3キーの例
1	source_data	データ抽出処理によって、Webから取得した直後のデータを格納するストレージを指す	XXXXXX（6桁）-YY（2桁） XXXXXXはグラフNo、YYは同一グラフで取得するデータの数を表す 対象ステークホルダーを3桁目の数値で細分化している 3桁目が「0」：産政局 3桁目が「1」：中企庁 3桁目が「3」：その他データ	000001-01/000001-01_202301.xlsx 000002-01/000002-01_202301.pdf 000003-01/000003-01_202301.csv 000003-02/000003-02_202301.csv
2	raw_data	データ整形処理によって、source_dataを行列が明確な形式に変換したtsvファイルを格納するストレージ	source_dataと同様	000001-01/000001-01_202301.tsv 000002-01/000002-01_202301.tsv 000003-01/000003-01_202301.tsv 000003-02/000003-02_202301.tsv
3	DWH	データ加工処理によって、raw_dataを加工したtsvファイルを格納するストレージ	ZZZZZZ（6桁） ZZZZZZは作成順に採番している	000001/000001.tsv 000002/000002.tsv 000003/000003.tsv
4	DWH（公開用）	オープンデータとして公開するDWHを格納するストレージ		
5	master	DWHの加工でマスタデータとして利用するtsvファイルなどを格納するストレージ	source dataと raw data の場合 No.1 のフォルダ配下にマスター用データを配置 DWH、DWH（公開用）の場合、No.3、No.4 のフォルダ配下にマスター用データを配置	000002-01/arrow1.jpg 000002-01/arrow2.jpg 000002-01/arrow3.jpg 000001/master.tsv

3.2 データ整備方法

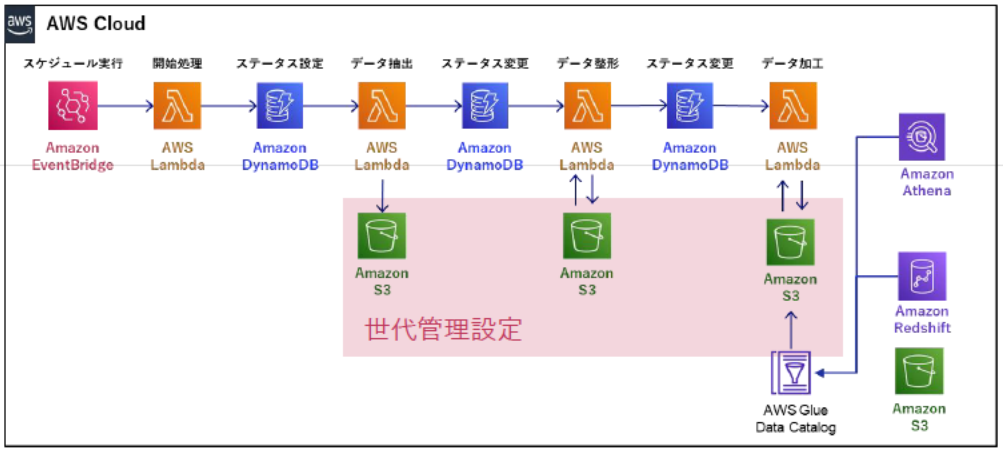


④ 基盤としての設定検討（定期更新）

- グラフの最新化のためEventBridgeとLambda、DynamoDB Streamsによってデータ整備を自動化する

No	処理名	AWS	概要
1	定期実行イベント	Event Bridge	cronで指定のタイミングで実行される Lambdaの定期実行処理を開始する。
2	加工・整形監視イベント	Event Bridge	加工・整形監視処理を15分間隔で開始するイベント
3	定期実行処理	Lambda	DynamoDBの抽出ステータスと整形ステータスを「未実行」に更新する その後、抽出ステータスを「実行中」に更新し、加工・整形監視イベントを起動する
4	加工・整形監視処理	Lambda	全データの抽出ステータスと整形ステータスが「実行済」か「異常終了」になっている場合、 加工ステータスを「実行中」に更新する
5	データ抽出処理	Lambda	各データの抽出を実行する 処理完了後、抽出ステータスを「実行済」か「異常終了」に更新する
6	データ整形処理	Lambda	各データの整形を実行する 処理完了後、整形ステータスを「実行済」か「異常終了」に更新する
7	データ加工処理	Lambda	各データの加工を実行する 処理完了後、加工ステータスを「実行済」か「異常終了」に更新する
8	データ抽出処理の実行	DynamoDB	抽出ステータスが「実行中」のデータ抽出処理を開始する
9	データ整形処理の実行	DynamoDB	整形ステータスが「実行中」のデータ整形処理を開始する
10	データ加工処理の実行	DynamoDB	加工ステータスが「実行中」のデータ加工処理を開始する

3.2 データ整備方法



④ 基盤としての設定検討（世代管理）

- データ破損時の復旧、データ分析時の再現性確保のため、S3のバケットにて世代管理を実施する

世代管理方針

- 同一ファイル名のデータについては世代管理を実施する
- 過去の世代のデータについては、「過去世代の保管期間」を超えた場合、削除する
  - No.1,2については、処理実行後1年間を過去世代保管期間とする
  - No.3,5については、処理実行後10年間を過去世代保管期間とする
  - No.4については、No.3の一部のデータを公開用として格納するため、No.3にて過去世代の保管を実施していることから、過去世代の保管は行わない

No	用途	S3バケット名		過去世代保管期間	現世代保管期間
		本番	開発		
1	SourceData	metidm-lake-prod	metidm-lake-dev	1年	永年
2	RawData	metidm-cage-prod	metidm-cage-dev	1年	永年
3	DWH	metidm-dwh-prod	metidm-dwh-dev	1年	永年
4	DWH(公開用)	metidm-opdata-prod	metidm-opdata-dev	1年	永年
5	MasterData	metidm-master-prod	metidm-master-dev	1年	永年
6	アップロード用	metidm-uploader-prod	metidm-uploader-dev	1年	永年

3.3 データ可視化方法

データ可視化方法は、大きく「データ連携」・「データ加工」・「グラフ作成」の3つの作業に分類することをベストプラクティスとした

データ連携

- AWS Athenaによって、DWHのデータをTableauに連携する

データ加工

- Tableau Desktopの機能を使ってDWHのデータをグラフ描画用に整形する

グラフ作成

- 加工したデータをグラフに描画する
- グラフの描画形式、データの範囲や軸の設定などを行う

3.3 データ可視化方法

① データ連携

設定した認証情報を用い、TableauからはAthenaのテーブルスキーマへ、PowerBIからはRedshiftのテーブルスキーマへ接続し、必要なDWHテーブルを使用する

連携設定

- Tableau
  - ap-northeast-1 リージョンを指定
  - 認証情報として、専用接続ユーザのアクセスキーとシークレットキーを指定
  - クエリログを保管するS3バケットを指定
  - データロードは「抽出」を使用（ライブは使用しない）
- PowerBI
  - Redshiftのワークグループ、データベース名を指定
  - 認証情報として、データベースユーザのユーザ名、パスワードを指定
  - データロードは「インポート」を使用（DirectQueryは使用しない）

3.3 データ可視化方法

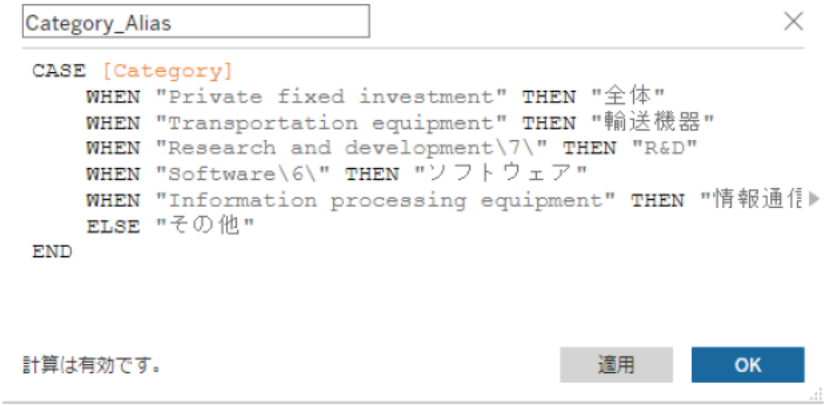
② データ加工

DWHテーブルのデータを用いて行う可視化に際し、表示に必要な情報追加やデータ属性変換を行う

加工処理例	概要
年月列追加	DWHテーブルでは年、月が別列のため、結合して日付型の年月カラムを追加する
データ列追加	DWHテーブルを結合する際、共通の名前を持つ列を追加する
表示用列追加	DWHテーブルで英字で格納されている項目名等を日本語化して表示用に利用する列を追加する
数値に変換	DWHテーブルでは文字列となっている列を数値に変換する
連続値に変換	グラフに表示する数値を不連続値から、連続値に変換する

年月列追加

“EN” という固定値をもつ「GEO」列追加



表示用列追加

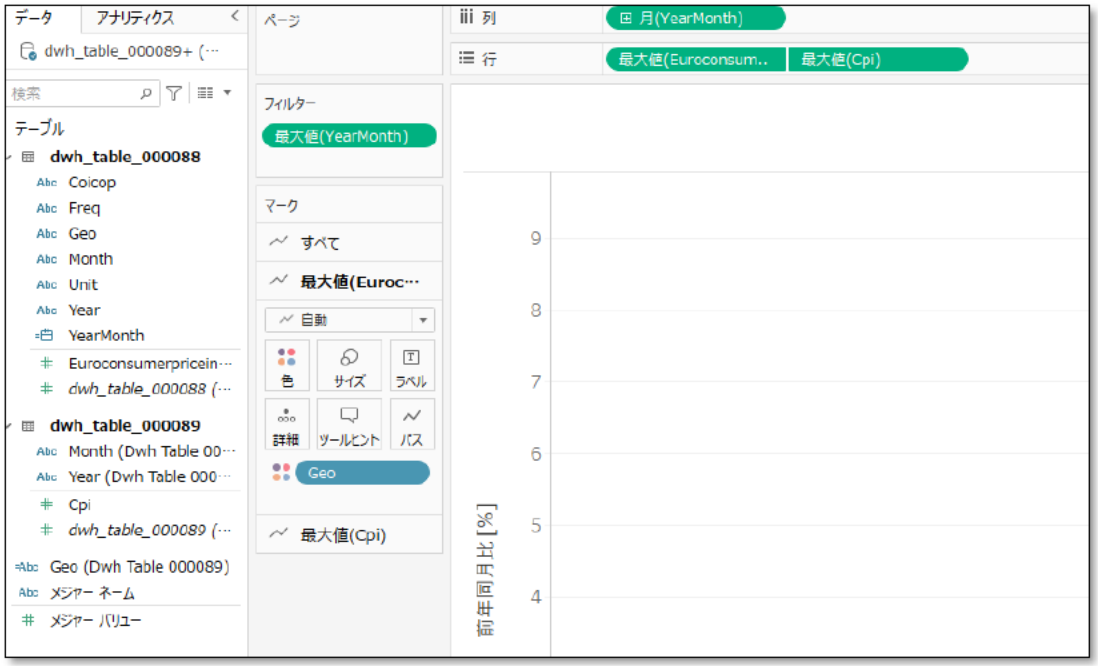
3.3 データ可視化方法

③ グラフ作成

日付情報と数値をもとにグラフを作成

連携設定

- 「列」に「年月」を指定し、「行」に表示する連続値を指定する
- 凡例に共通の列を指定する
- その他、グラフ作成要件に従い、作成する



4. データ整備・可視化方法の標準化検討

4.1 接続・収集方式における作業
4.2 編集方式における作業
4.3 加工方式における作業
4.4 データ加工処理概要
4.5 セキュアコーディング
4.6 グラフ作成
4.7 定期的な実行

4.1 接続・収集方式における作業

既存のデータ抽出関数で新規データを抽出する場合、以下の作業が必要

手順	作業内容	作業詳細
1	データ・グラフ一覧およびデータ抽出機能処理設計書の更新	データ・グラフ一覧に対象データを採番し追加する 抽出機能処理設計書に対象データと対象関数をデータ一覧シートに追加する
2	URLの確認およびフォーマット検討	URLをアクセスできることを確認し、ダウンロードしたファイルの拡張子を確認する URLが年月日に応じて動的に変化する場合、URLのフォーマット形式を検討する
3	DynamoDBの設定	1, 2の手順で採番、確認した項目をDynamoDBに設定する
4	Lambdaのテスト	DynamoDBからLambdaの実行を開始させ、 抽出ステータスが2になることを確認する
5	テスト結果確認	4の実行でダウンロードされたファイルが想定通りであることを確認する

4.2 編集方式における作業

新規データを抽出した場合、データ整形処理の新規作成が必要

手順	作業内容	作業詳細
1	データ・グラフ一覧の更新	データ・グラフ一覧に対象データを採番し追加する
2	整形機能処理設計書の更新	整形機能処理設計書に対象データと新規関数をデータ一覧シートに追加する 新規データの整形後データ概要のシートを追記する
3	Lambda関数の作成	3で作成した設計を元に新規関数を作成し、関数番号を採番した後、 Lambda関数にデプロイする
4	DynamoDBの設定	1, 2, 3の手順で採番、確認した項目をDynamoDBに設定する
5	Lambdaのテスト	DynamoDBからLambdaの実行を開始させ、 整形ステータスが2になることを確認する
6	テスト結果確認	5の実行でダウンロードされたファイルが想定通りであることを確認する

4.2 編集方式における作業（注意事項）

以下のようなパターンにおいて、既存のデータ整形処理ではデータが整形できなくなる事象が発生する

1. 取得データのフォーマットが変更される（行列の位置ずれ等）
2. 取得データのファイル拡張子が変更される（例：Excel⇒PDF形式）

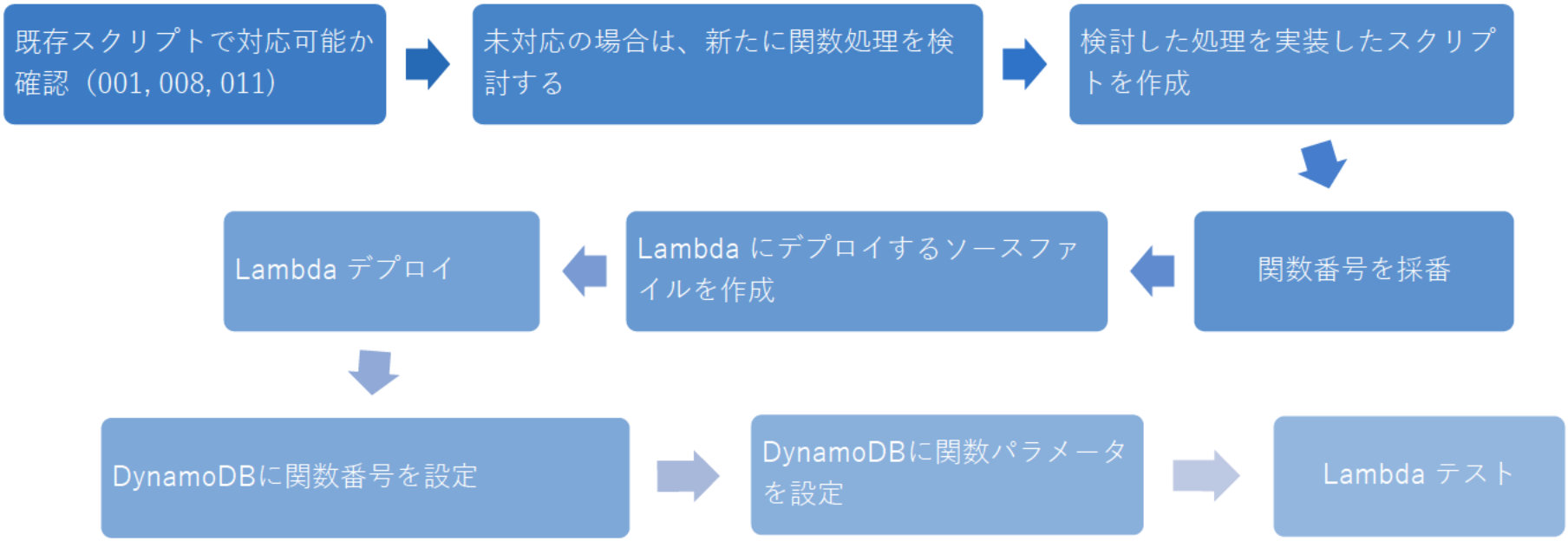
その場合、既存のデータ整形処理を修正する必要がある

手順	作業内容	作業詳細
1	変更の仕様確認	データの変更の詳細を確認する データの変更の期間（一時的か恒久的か）を確認する
2	Lambda関数の修正	データの変更内容に合わせて関数の修正を行い、Lambdaへデプロイを実施する なお、DynamoDBの設定変更のみで対応可能な場合、本手順をスキップする
3	整形機能処理設計書の更新	2の関数修正で設定書に修正が必要な場合は、設計書の修正を行う
5	DynamoDBの更新	1, 2, 3の手順で更新した項目をDynamoDBに設定する
6	Lambdaのテスト	DynamoDBからLambdaの実行を開始させ、 整形ステータスが2になることを確認する
7	テスト結果確認	6の実行でダウンロードされたファイルが想定通りであることを確認する

4.3 加工方式における作業

新規DWHの作成が必要となる場合、既存の加工処理の組み合わせ流用で作成、
場合によっては新規の加工処理を一から作成する

- 新規DWH固有の集計処理が必要となる場合
  - Tableau などのBIツールで集計できるか確認する
  - BIツールでは容易に処理できない場合、集計用関数を定義する
- 複数のRawDataを結合するが既存のスクリプトでは対応できない場合
  - 既存スクリプト008、011を参考に、結合処理をする関数を新たに定義する



4.4 データ加工処理概要

lambda_function.py

lambda_handler 関数の概要

- DynamoDB から設定を取得し辞書にする
- 辞書の dwh_func_no の番号を元に呼び出したい関数をインポートする
- 関数の引数は、S3リソースと、設定値辞書とする
- インポートした加工処理関数を呼び出す
- 加工処理から返却されるヘッダリストを用いてAthenaテーブルを作成する
- 加工ステータスを更新する

インポートする加工関数の概要(001)

- 関数の引数は、S3リソースと、設定値辞書とする
- RawData のDataFrameを取得する
- 設定されている単位合わせが必要なカラムに対し、設定した単位で単位合わせを行う
- S3 に格納する
- pandas.DataFrame のカラムリストを返却する
- エラー時は空のリストを返却する

4.4 データ加工処理概要（続き）

- インポートする加工関数の概要(008)
  - 関数の引数は、S3リソースと、設定値辞書とする
  - 辞書に設定されたRawData情報分ループする
    - RawData のDataFrameリストを取得する
    - 取得したリストを縦結合する
    - 結合キー以外のカラムを数値に変換する
    - 当該RawDataの中で、ソートキーカラムでソートする
    - 当該RawDataの中で、重複排除対象カラムを使い重複排除する
    - リストAに完成したRawDataのDataFrameを追加する
  - 前述のリストAのDataFrame 分ループする
    - inner 結合する
  - 設定されている単位合わせが必要なカラムに対し、設定した単位で単位合わせを行う
  - S3 に格納する
  - pandas.DataFrame のカラムリストを返却する
  - エラー時は空のリストを返却する

4.4 データ加工処理概要（続き）

- インポートする加工関数の概要(リストで取得したDataFrameに独自処理を行う場合)
  - 関数の引数は、S3リソースと、設定値辞書とする
  - RawData のDataFrameリストを取得する
  - pd.concat で縦結合する
  - ソートキーでソートする
  - 重複排除する
  - 独自処理を行う（例：いくつかの列から別の数値を計算して新たな列を作る）
  - 設定されている単位合わせが必要なカラムに対し、設定した単位で単位合わせを行う
  - S3 に格納する
  - pandas.DataFrame のカラムリストを返却する
  - エラー時は空のリストを返却する

4.4 データ加工処理概要（続き）

共通関数一覧

関数	処理概要
get_dwh_info(table, dwhno)	対象のDWHNo. の情報を取得し辞書を返却する table は DynamoDB のマスターテーブル
put_data_frame(s3, bucketName, targetNo, fileName, df, encoding="utf-8")	DataFrame を対象バケットの DWHNo. 配下に格納する
get_data_frame_list(s3, bucketName, rawNo, encoding="utf-8")	S3 上対象バケットの RawNo. 配下にある TSV を全て DataFrame にしてリストで返す 重複排除は呼び出し元で実装する
get_data_frame(s3, bucketName, rawNo, sortcols=None, ascending=None, subset=None, encoding="utf-8"):	S3 上対象バケットの RawNo. 配下にある TSV を全て結合した DataFrame で返す 重複排除を行う
create_athena(athena, athenaDb, bucket, dwhno, columns, athenaQueryLogs)	Athena テーブルを作成する関数
drop_athena(athena, athenaDb, dwhno, athenaQueryLogs)	Athena テーブルをドロップする関数
update_status(table, **kwargs)	DynamoDB ステータスを更新する関数
force_to_numeric(frame, raw_subset)	数値として扱うカラムに対して数値に変換する関数

4.5 セキュアコーディング

AWS Lambda で起こり得る脆弱性に対して実施した対応については下記の通り。
※グレーアウト部分は対処方法に記載の理由で今回対象外とした。

サーバレスに関する脆弱性top10（OWASP Top10 Serverless）	脆弱性	脆弱性の概要	分類	対応要否	今回対応可否	今回の対処方法	対応状況
A1:2017 Injection	OS Command Injection	環境変数の内容を取得するコマンドの悪用による IAM Role のクレデンシャルの奪取	コーディング	必要	○	OSコマンドの実行は、tabula-java のみに限定する。 書き込む、読み込むファイルのローカルパスには、オリジナルファイル名を用いないようにする。また、不要な制御文字を含まない形式にする	済
A2: Broken Authentication	Broken Authentication	認証の不備	基盤	必要	○	ユーザ認証を確実に行う。不要なIAMユーザは登録しない。 基盤構築上の課題であるため、セキュアコーディングの対象外とする。	
A3：Sensitive Data Exposure	Sensitive Data Exporsure	機密データの露出	基盤	必要	○	機密データは、S3 に格納しない。 S3 のアクセス制限を設定する。外部公開しない。 基盤構築上の課題であるため、セキュアコーディングの対象外とする。	
A4：XML External Entity	XML External Entity (XXE)	XML外部実態参照 Path Traversal によるファイル内容の流出など	コーディング	必要	○	lxml の場合、resolve_entities=Falseを設定する 標準xmlは、XXEはデフォルトOFF	済
A5: Broken Authentication	Broken Access Control	アクセス制御の不備	基盤	必要	○	IAMでのロール、ポリシーを適切に設定する	
A6: Securit Misconfiguration	Security Misconfiguration	セキュリティ設定のミス	基盤	必要	○	S3, DynamoDB, Athena, Lambda などのセキュリティ設定を 行う。 基盤構築上の課題であるため、セキュアコーディングの対象外とする。	

参考URL
https://docs.aws.amazon.com/ja_jp/lambda/latest/operatorguide/execution-environment.html
https://owasp.org/www-project-serverless-top-10/

4.5 セキュアコーディング（続き）

AWS Lambda で起こり得る脆弱性に対して実施した対応については下記の通り。
※グレーアウト部分は対処方法に記載の理由で今回対象外とした。

サーバレスに関する脆弱性top10（OWASP Top10 Serverless）	脆弱性	脆弱性の概要	分類	対応要否	今回対応可否	今回の対処方法	対応状況
A7: Cross-Site Scripting	Cross-Site Scripting	クロスサイトスクリプティング	コーディング	必要	○	スクレイピング時に、javascript を実行しない。	済
A8: Insecure Deserialization	Insecure Deserialization	任意コードの実行による環境変数から IAM Role のクレデンシャルの奪取	コーディング	不要	×	pickle など、シリアライズモジュールは今回使用予定がないため対処不要	
A9: Using Components with Known Vulnerabilities（	Using Components with Known Vulnerabilities	既知の脆弱性を持つコンポーネントの使用	運用	必要	○	Lambda レイヤーに含めるパッケージは、定期的に更新する。 また、使用しているパッケージに脆弱性が報告された場合についても、更新パッケージを反映する。といった運用時の課題であるため、今回のセキュアコーディングの対象外とする	
A10: Insufficient Logging & Monitoring	Insufficient Logging & Monitoring	不十分なロギングとモニタリング	コーディング	必要	○	CloudWatch Logs などの設定を適切に行う。 本プロジェクトにおいては、ログ出力とエラー検知のコーディングを対応しているため、問題ないと考える	済
Other	Server Side Request Forgery (SSRF)	Lambda runtime API などにアクセスすることによる機微な情報の奪取	コーディング	必要	○	不要なサイトにアクセスしないようにする ・DynamoDBに設定した取得先URLのサニタイジング（実装時に取得サイト確認済み） ・XML,HTML,PDF,Excelなどから取得したURLにはアクセスしないようコーディングする	済
Other	Remote Code Execution (RCE)	Lambdaやパッケージなどの脆弱性により、任意コードの実行による環境変数から IAM Role のクレデンシャルの奪取	運用	不要	×	Lambda レイヤーに含めるパッケージは、定期的に更新する。 また、使用しているパッケージに脆弱性が報告された場合についても、更新パッケージを反映する。といった運用時の課題であるため、今回のセキュアコーディングの対象外とする	
Other	その他	–	コーディング	必要	○	Excel マクロを解釈しないよう pandas の read_excel を使用する。	済
Other	その他	–	コーディング	必要	○	Lambda で tabula-py を実行時など、/tmp などに配置したリソースは削除する	済

参考URL
https://docs.aws.amazon.com/ja_jp/lambda/latest/operatorguide/execution-environment.html
https://owasp.org/www-project-serverless-top-10/

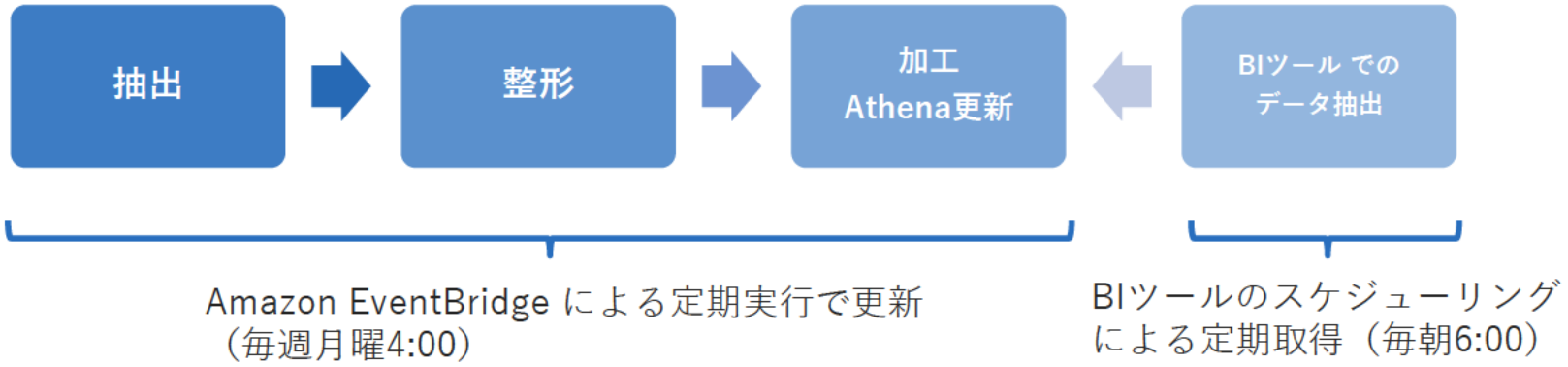
4.6 グラフ作成

複数のDWHを使用する場合、年、月などをキーに結合する

4.7 定期的な実行

グラフの最新化のため、定期更新は以下の流れで実現

- 定期的に実行される Lambda の加工処理内で、AWS Athena の更新を行う
- BIツールにパブリッシュされたダッシュボードはスケジュールによりデータを更新する




Orchestrating a brighter world
NEC